It takes a winner to take his share

Suleyman Malki[1], Lambert Spaanenburg[1] and Berend-Jan van der Zwaag[2]
Lund University / LTH
P.O.Box 118, SE-22100 Lund (Sweden)
Phone: +46 (0)46 222 4931
Fax: +31 (0)46 222 4714
Twente University / EWI
P.O.Box 217, 7500 AE Enschede
Phone: +31 (0)53 489 2094
Fax: +31 (0)53 489 1060
E-mail: {suleyman|lambert}@it.lth.se; b.j.vanderzwaag@utwente.nl

Abstract — **Novelty detection is based on the creation of a space with similarity metric. It is discussed that the design of a neural detector is a compromise between promptness, universatility, robustness and sensitivity. The feed-forward topology is chosen from three alternatives for its ability to design that compromise by applying structural redundancy. A generic FPGA implementation supports the use in adaptive intelligent systems.**

Keywords— **Novelty detection, HW/SW Co-design, Field-Programmable Gate-Array, Autonomous systems, Similarity metric**

I. INTRODUCTION

The application of structural measures to enhance robust behavior is a fundamental engineering concept. By coupling the result of an action back to the origin, the control can be adapted to provide a more stable cause-effect relationship. Conversely, the same can be reached by anticipating the result of an action along the forward path. These two measures have to be carefully analyzed in the different applications as stability is not guaranteed by nature, and may even a disguise for chaotic behavior. Next to studies in system theory, we see a similar interest in a range of engineering and other disciplines.

Abnormalities can be caused by shifts in the relation between the viewer and the subject. To allow for an objective view, we assume that the observation platform is stable beyond any doubt. Further we will discard any external influence. In all other cases, abnormalities find their root in the functioning of the observed system. This does not label it as good or bad, because the modeled expectance of the observation may be incomplete. Only in the presence of a perfect model will the novelty indicate a potential malfunctioning. This world where any aberration can be formulated as a modeled presence of a fault is where the classical Fault Detection & Isolation is focusing on. We will keep some more distance and look into means to distinguish whether a process is structurally becoming different from the previously established model.

Novelty detection is usually at odds with system robustness. Large abnormalities tend to be visible for reason of their clear consequences. The small ones are of a larger concern, because early detection could help to take timely counter measures and thereby prevent the imminent catastrophe. To detect small novelty measures takes a large sensitivity. Sensitive systems do unfortunately often lack robustness or even stability.

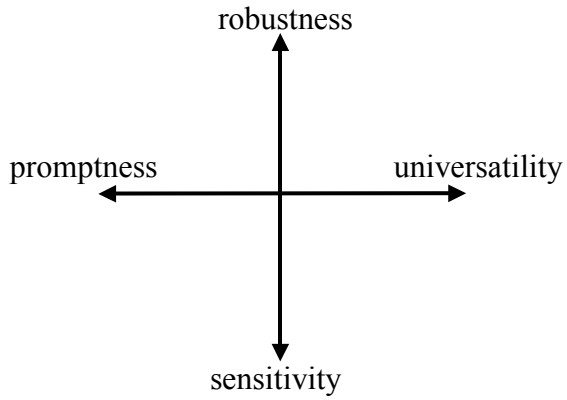

Figure 1: The parameter space for novelty detection and isolation (NDI).

This is shown vertically in Figure 1. The other consideration in this figure shows the balance between classification speed and the ability to handle real novelties: abnormalities that have not been observed before. Because a balance between conflicting demands seems to rule both dimensions in this figure, the ideal situation will reflect a circle in the depicted plane. As pointed out in [1], the promptness of the response is a process oriented feature while the universality is more a model characteristic. This indicates that a prompt but universal novelty handling scheme can best be achieved

by developing independent models for the process and for the novelty detection. This paper follows that lead and shows some techniques to achieve both sensitivity and robustness.

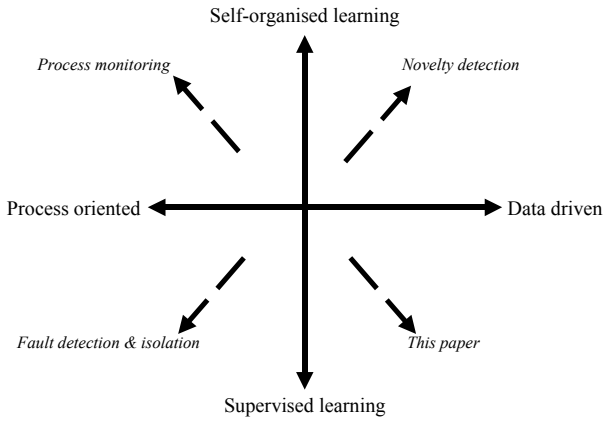

Figure 2: Use of learning methodology in several areas of detection and isolation [1].

We will first discuss the role of feedback in the establishment of robustness for neural structures. Subsequently the issue of abnormality detection is raised. In section III robust abnormality detection is studied in terms of physical analogons to show how quality classifiers can be obtained. Finally the new scheme is evaluated for its usage in HW/SW in-line quality assurance.

II. The Role of Feedback

The Self-Organizing Feature Map (SOFM) is based on the principle of the N-flop. Within a geometrical region of interest (often the nearest neighbors within a specified radius) the neurons are mutually connected in a dominating scheme. Eventually the winner takes all. In extension to the 2-flop (or more commonly called flip-flop) we find for the N-flop the cross-connection of NOR-gates such that the highest input will force all other gates to close. The corresponding gate will output a '0' while all others give a '1'. The weak point of this approach is that by mapping an n-dimensional problem onto a 2-dimensional network a problem coding may occur for which no winner can be found. In such a case, the network gives no result and can also not be debugged on basis of erroneous behavior. Another weak point (sometimes quoted as an advantage) is the lack of supervision, which makes quality hard to control.

The Cellular Neural Network (CNN) is based on the coupling scheme that underlies the Chua coupled oscillators. The basic difference is that the local computation converges in a limited amount of cycles. Two templates govern this: one for the co-operation on basis of nodal input values and one for the co-operation on basis of nodal output values. This forces the interaction between neighboring nodes to be of a defined character. Most applications are in image processing, where already templates have been discovered for many simple and more complex operations.

On first sight, there appears to be no feedback in feed-forward networks. The network performs a straight function mapping without any dynamic aspect. Some form of feedback is however available through the learning function, causing any discrepancy between presented example and expected result to produce an adaptation of the synaptic weight values. But when the learning has ended, this feedback arrangement is cut and outliers may still cause an abnormal reaction.

For a long time, the characteristic difference between biological and artificial neural networks was the lack of redundancy. Redundant nodes may in turn cause stability and are therefore a desired design aspect. In fact, redundancy can be seen as the feed-forward counterpart in terms of robustness measures. A typical result of redundancy can be seen in Figure 3. The demands for computational precision can be appreciably lowered by the judicious introduction of redundant nodes.

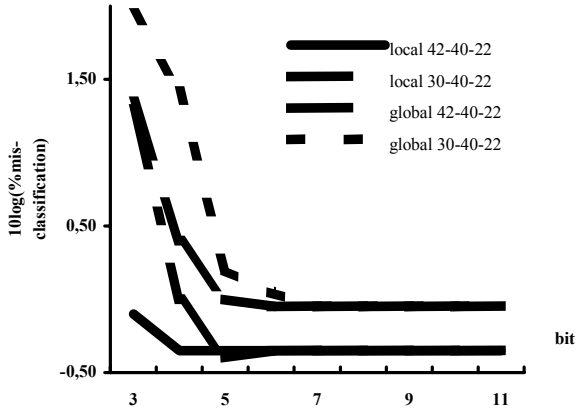

Figure 3: False classification rates for license-plate recognition [2].

This brings us to the actual focus of interest of this paper: how does redundancy act in establishing the desired robustness and how can it be used to achieve the balance between robustness and sensitivity that makes a good classifier? The goal is to find a methodology that enables us to balance the trade-off during design. The SOFM lacks that flexibility, while for the CNN robustness is still being researched [3]. As we will see later in this paper, the feed-forward provides this capability but in a not understood manner.

III. ABNORMALITIES

Neural networks are trained by examples; their knowledge will therefore reflect this history. It can be stored in two ways: either (a) as a cluster in the n-dimensional space for points that share enough features, or (b) as a set of dividing vectors. The training method is based on interpolation. Any unknown example is brought into accordance with the stored ones by slightly adapting the synaptic weights that define the dividing vectors. In case of an occasional novelty, the adaptation will only be slight and will soon be nullified when known examples persist in their training demand. When the novelty is chronic, the training will attempt to compromise. Consequently a compromise is constructed between essentially two different phenomena.

In Figure 4 this is shown for the case of the classification of (non-) occluded printed characters. When only full characters are learnt, they are adequately remembered but hardly recognized when the occlusion becomes larger than 15%. Vise versa is true, when the occluded characters are learnt but tested with full characters. When both sets are trained simultaneously, they will both be remembered but al a lower success rate.

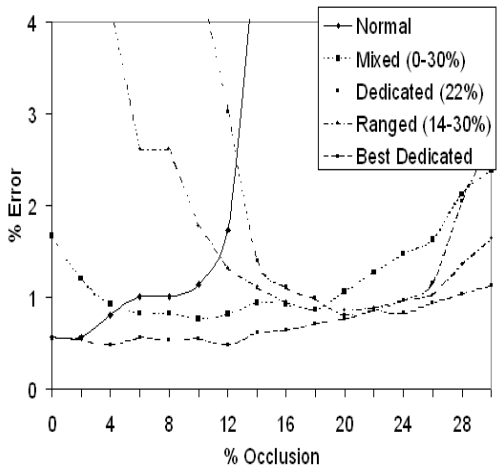

Figure 4: A comparison of the error rate for various neural classifiers built using different training sets [4].

Care must be taken when the outlier presents a situation outside the learned interval. As little has been learned before, the outlier will be rapidly accommodated. In fact, being the only representant of the outer world, there is no way to discern it from the previously captured model.

One way to solve this problem is by using the concept of 'confidence': neural network is not only trained to give an answer but also to supply a value that tells how confident it is of the correctness. This does not have to be a simple error measure but can be any measure of similarity. The confidence measure is of special advantage where neural sub-networks are combined into a single modular network. For instance, in an experiment on surface inspection, a 3-tier network has been used to combine model pre-knowledge to detect potential scratches.

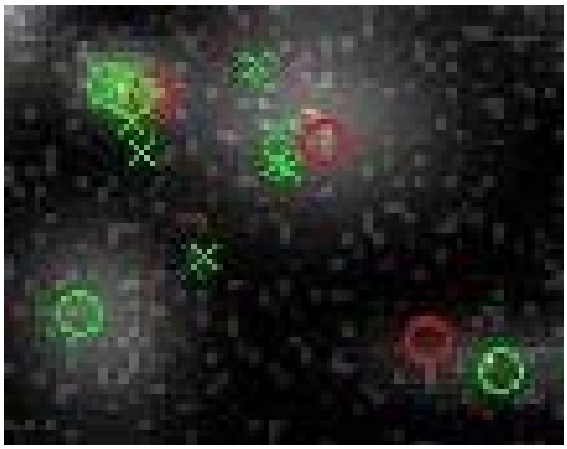

Figure 5: Spot detection with confidence.

This confirms our earlier suspicion that the feed-forward network is better equipped to handle abnormalities, once it is trained to be robust. Therefore the following presents a discussion on the nature of robustness by means of a physical analogon.

IV. A BI-VALENT ANALOGON

The success of neural abnormality detection depends strongly on the ability to model normality. On first sight, this may seem an impossible task as for several reasons the classification will never be accurate. This is true but only in a numerical sense. By the curvature in the error space, a sample will almost never be classified as 100% correct. In a functional sense, the negative expectation is false. Close to 100% is functionally enough for membership as long as cluster separation is maintained.

This principle rules universally in engineering. Lets look at the clustering of analog voltage values at the output of a logic gate. Though the "high" voltage levels will hardly ever be equal to "Supply" and the "low" voltage levels will hardly ever be equal to "Ground", the cluster based detection by a next gate will not see the difference between the many "high" levels or between the many "low" levels. Further, and even more crucial, is the existence of a steep transition between "high" and "low", where voltage levels are either not accepted by design or can never be stable by construction. In the following we will briefly discuss this by analogy.

A. *The mass analogon*

The first observation is by looking at the Earth/Moon gravity system. Objects within the attractive force field of the Earth will only reach the Moon when the velocity is high enough to be non-stable. Vice versa, objects that leave the Moon with a too low velocity will never reach Earth.

Attraction forces obey the generic mathematic expression with k being some constant, m and M attributes of respectively the attracted and the attracting body (e.g. mass, heat capacity or electrical charge), and r the distance between the bodies. An object between two attracting bodies will move towards the one with the largest force, i.e. somewhere between the two bodies the total force will change sign and an object will move away in the direction for the sign. The point where the total force changes sign, is characterized by having a zero force: a meta-stable point in which the body will not move, but for any infinitesimally small displacement it will immediately leave in the direction of the field.

The force field imposes a metric on the problem space by building a relation between the distance in space versus unit steps in the field and likewise between the distance in force measure unit steps in the problem space. The relation is nonlinear as the double integration of the forces yields a natural logarithm. Consequently, the mass analogon seems easy to be implemented on a neural structure. However, it is not clear what the equivalence of an object with mass can be. But inspired by the mass analogon a broad diversity of so-called clustering algorithms have been devised, also in the neural domain. Such algorithms aim to combine data elements into two or more classes based on a mathematical formulation for attracting and repelling forces.

B. *The fluid analogon*

The major drawback of the mass analogon is the functional character, i.e. there is a uni-valued force field with a singular ridge for those places where the field changes sign. Such a scheme allows to create a high sensitivity but robustness is not supported in a transparent manner. Around the meta stability the field is small but unequal zero. Any object in this region will slowly but unstoppable drift towards some existing cluster.

A different situation occurs when the problem space is filled with many small particles of an either positive or negative attribute. The intrinsic space is filled with the material in an even spread. This neutralizes the individual contributions within the overall effect. In fact, a natural tendency for global neutrality might be discerned where any disturbance will be disheveled into the natural equilibrium.

The problem space might be personalized by local aberrations from the global equilibrium, imposed by some external source. The effect might be called function or structure, signifying the specificity as intended. For the feed-forward neural structure, this can be implemented through the use of structural redundancy. This makes redundancy an important means for influencing the robustness of the design without meliorating the sensitivity.

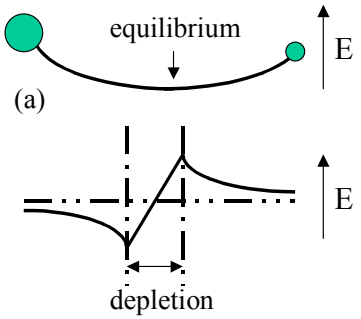

Figure 6: Attraction fields according to (a) the mass and (b) the fluid analogon.

V. IMPLEMENTATION ASPECTS

Digital neural hardware has not enjoyed much popularity in the past. The many synapses define a complex wiring scheme, which can only be simplified by temporal encoding [5] or multiplexing [6]. Further the sheer size of the multiplier, on which the synapse is built, is a concern, that seems to necessitate for temporal iteration on a limited amount of resources.

It has been suggested, that a spatial computing style would be appropriate. Unfortunately, microelectronic technology could only support such a realization as a board of single neuron ASIC's [7]. But time has passed and meanwhile hardware complexity has become a lesser restriction. Recent FPGA architectures provide a large amount of Configurable Logic Blocks with interspersed optimized RAM and multiplier macro's.

Another point worth noticing is the maturization of reconfiguration technology as a kind of in-line programming style. In [8] such FPGA aspects are proclaimed as the new carriers of innovation. It is the aim of this paper to find whether this also has impact on the realization of artificial neural networks.

The simplest realization of a neuron with incoming synapses is based on a multiplying adder. Many simultaneously active neurons can be imitated by

executing the single instance for the many signal settings. Such different settings can be allocated in SRAM macros (Figure 7a).

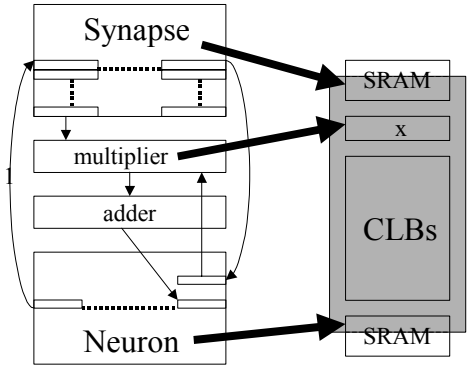

Figure 7: (a) Neural and (b) FPGA Node.

The Neuron Value Store NVS contains the axon value and the start address in the Synapse Value Store SVS where the incoming synapses are administrated. The SVS contains the NVS address of the sourcing neuron, the weight value and the address of the next synapse. The bias can be either a value for a neuron or a synapse with no source.

The 18k bit dual-port SRAM can be generated in various depth and width configurations. Design considerations, as described in [2], allow the values to be limited to 8 bits. Figure 3 gives an example. The structural information on normal-size networks can then be stored in maximal 100 words of 16 bits, leaving ample room for supporting functions.

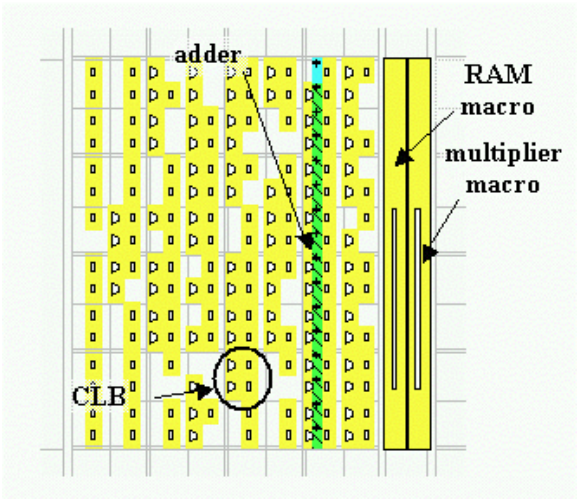

Figure 8: Module floor plan

A. *Modular Neural Network*

For a typical FPGA architecture, the chip is divided into super blocks (Figure 7b). For the purpose of the later discussion, we have shaped the neural module on a super block such that the SRAM macro's are shared between consecutive networks to pass the values. A typical module floorplan appears in Figure 8.

Modular neural networks or more commonly called multi-nets are combinations of several neural networks [9]. They are of growing interest, as the implied feature redundancy is believed to make the overall net more accurate than the parts. Moreover, multi-nets can be easier to understand and to modify.

From the observation that, with growing problem size, the monolithic network has increasing difficulty to learn with sufficient quality [10], one may expect not better from a multi-net. In both cases, the learning process suffers from the entropy in the example set. This can only be resolved (a) by data preprocessing, (b) by inclusion of pre-knowledge or (c) by domain structuring. Such can be achieved by using modular networks [11]. The claim that more than 80% of the development time for monolithic networks is spent on the data preprocessing underlines this observation [12].

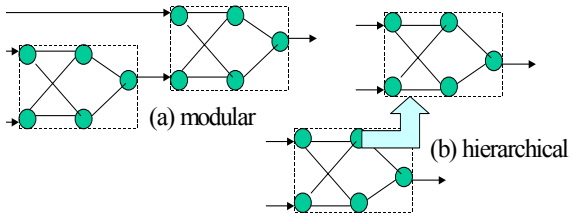

Figure 9: Some neural network types.

A modular network can be interpreted as a multi-layer hierarchical network where ultimately on the highest level the weights are constant and equal to one (Figure 9a). In other words, the top modular composition has lost its exclusive neural outlook and has become heterogeneous in nature by allowing for components of any fabric. By the expansion to multiple layers, and adding weights on the connections between networks, hierarchy is enabled as depicted in Figure 9b.

What is similar to both types of networks is that both hierarchical and modular networks apply functional specialization, although in a different form. Specialization enables the fusion of existing knowledge into the neural network, as was shown for modular networks in [14].

B. *Spatial neural computing*

The temporal design of a module contains already all the necessary ingredients for neural data processing. Scaling can easily be achieved by increasing the network representation within the SRAM, but this will soon lead to unwielding long execution times. The alternative is the replication of the elementary module over the chip. With the coming of FPGA devices like the Xilinx

Virtex-II family this option has become very real.

The added advantage of spatial computing is the opportunity to learn the modules of a neural network almost in parallel. This was first noted in [15] in the analysis of the unlearning potential of neural composition. When a network is assembled from trained and empty modules, it frequently happens that the inserted knowledge is swept away during the first epochs and the network continues as if nothing had been there.

The remedy has proven to be a time-ordering of the activation time of the individual modules [15]. The original circuit was next to impossible to learn, but already small delays between the activation of the modules brings learning time back to acceptable properties, wherein the overall network is trained in just slightly more time than a single module.

As all inputs and outputs of the modules are handled by using the SRAM, passing information between modules views the SRAM as a blackboard. It is not necessary to signal new events by semaphores as a neural network is robust enough to allow for the occasional mixture of old and new values [16].

VI. Discussion

Such considerations make for a compact arrangement by mere concatenation of the modules. Figure 10 shows the floor plan of such a spatially unrolled neural network. The design is behaviorally constructed in VHDL using Xilinx WebPACK 4.2i and simulated by ModelSim XE5.5. The Xilinx Core Generator allows generating the multiplier & RAM as facilitated by the chip. Then the logic synthesizer creates a mapping on the CLBs. Some manual intervention is required to confine the Place & Route to the envisaged area. Overall this results in the design shown in Figure 8, that uses 83 % of the available flip flops and 57 % of the LUT's.

The only real variable is the RAM usage, which in turn relates to the network size. This leads to using the average speed per line of RAM code (loc) as Figure of Merit. For our design this leads to 40 ns/loc. For a temporal design this number would be a constant, but for the spatial design the number of modules in the longest path divides the number. When compared to a temporal software design on a Pentium-III, the acceleration is by a factor 20, as earlier reported in [8].

By its modular structure, the network allows for in-product adaptation of the amount of redundancy. This facilitates tuning the classifier during the life time of an intelligent sensor. When the process model changes because of wear or ageing, the classifier can be adjusted without consequences for the floor plan. For similar reasons, the module is usable for built-in HW/SW test technology.

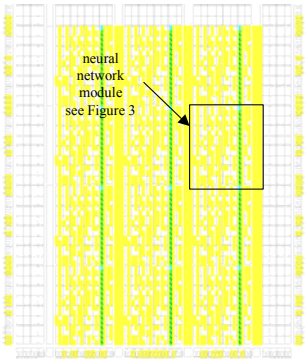

Figure 10: Floor plan of a spatial neural network

References

[1] M. vanVeelen, Ph.D. thesis, Groningen University (2004).

[2] L. Spaanenburg et al. "Training neural nets for small word width", Proceedings AmiRA'01 (Paderborn, October 2001) pp. 171 - 180.

[3] S. Xavier de Souza et al., "Automatic chip-specific CNN template optimization using adaptive simulated annealing", Proceedings ECCTD, Vol. II (Krakow, 2003) pp. 329-332.

[4] J.A.G. Nijhuis, A. Broersma, and L. Spaanenburg, "A modular neural network classifier for the recognition of occluded characters in automatic license plate reading", Proceedings FLINS2002 (Ghent, September 2002) pp. 363-372.

[5] A. Jahnke, U. Roth and T. Schoenauer, "Digital Simulation of Spiking Neural Networks", in: Pulsed Neural Networks, edited by W. Maas and C.M. Bishop, MIT Press, 1998.

[6] P. Richert et al., "ASICs for prototyping with pulse-density modulated neural networks", pp. 125 – 151, in: VLSI Design of Neural Networks, edited by U. Ramacher and U. Rueckert, Kluwer Academic Publishers, 1991.

[7] J. Quali et al., A customizable neural processor for distributed neural network, VLSI (1991) pp. 167 – 176.

[8] A. de Hon, "Reconfigurable Architectures for General-Purpose Computing, AI Techn. Rpt 1586 (MIT, Cambridge) 1996.

[9] A. Sharkey, "Multi-Net Systems", in "Combining Artificial Neural Nets" Ed. A Sharkey, Springer-Verlag, London, 1999, ISBN 1-85233-004-X.

[10] W.G. Macready, A.G. Siapas, and S.A. Kauffman, "Criticality and parallelism in combinatorial optimization", Science, Vol. 271, pp. 56-59 (1996).

[11] T. Caelli, L.Guan and W.Wan, "Modularity in Neural Computing", Proceedings of the IEEE 87, No.9 (September 1999), pp. 1497-1518.

[12] B. Schuermann, "Applications and Perspectives of Artificial Neural Networks", VDI Berichte, Vol. 1526, pp. 1-14 (2000).

[13] A.J.W.M. ten Berg and L. Spaanenburg, "Considerations of the Compositionality of Neural Networks", Proceedings ECCTD, vol. III (Helsinki, September 2001) pp. 405-408.

[14] L. Spaanenburg, "Over multiple rule-blocks to modular nets", Proceedings 23rd Euromicro (Budapest, 1997) pp. 698 – 705.

[15] R.S. Venema and L. Spaanenburg, "Learning feed-forward multi-nets", ICANNGA'01 (Prague, 2001) pp. 102 - 105.

[16] J.A.G. Nijhuis et al., "Delay-insensitive learning in a feed-forward neural network", Proceedings INNC'90, Paris (France) July 1990.